1. Write a C program to do matrix addition, matrix subtraction and to extract the min, max and average of the matrix elements

Your program should:

- Get two 4x4 matrices from the user
- Print the result of matrix subtraction, matrix addition
- Get a 4x4 matrix from the user
- Print the max (largest element), min (smallest element) and the average (sum of elements/number of elements) of all the elements

Your code should have the following functions:

**a. void GetMatrixElements( int matrix[4][4] )**

This function should get 16 values from the user and store it in matrix. The values are to be scanned in one row at a time i.e. for the 4x4 case, the 5th value entered by the user would be the value at row:1 column:0 (row and column values start at 0).

Example: To get the 4x4 matrix

```
1  2  3  4
5  6  7  8
9 10 11 12
13 14 15 16
```

Your function should produce an output like the following:

```
Enter row 0 col 0 value : 1
Enter row 0 col 1 value : 2
Enter row 0 col 2 value : 3
Enter row 0 col 3 value : 4
Enter row 1 col 0 value : 5
..........................
Enter row 2 col 3 value : 12
Enter row 3 col 0 value : 13
Enter row 3 col 1 value : 14
Enter row 3 col 2 value : 15
Enter row 3 col 3 value : 16
```

**Note:** Output of GetMatrixElements shown above, shows only the first 5 and last 5 elements, of the input matrix, your output should show this for all values. i.e your function should prompt the user to enter all the 4x4 values.
b. *void PrintMatrix( int matrix[4][4] )* 

The PrintMatrix function prints the elements of *matrix*. Elements belonging to the same row must be printed in a single line and should be spaced using a tab ( \t ). Each row has to be printed in a different line.

Example: For the 4x4 matrix shown in the previous example, your PrintMatrix function should produce an output similar to this

```
1  2  3  4
5  6  7  8
9 10 11 12
13 14 15 16
```

c. *void AddMatrix( int matrix1[4][4], int matrix2[4][4], matrix3[4][4] )* 

Your AddMatrix function should take in three 4x4 matrices as parameters, and store the sum of the first two matrices into the third.

Example:

If

```plaintext
matrix1 = 
1  2  3  1
2  3  4  1
4  5  6  1
1  1  1  0
```

```plaintext
matrix2 = 
4  5  6  1
2  2 -1  1
2  0  1  1
1  1  1  0
```

then

```plaintext
matrix3 = 
5  7  9  2
4  5  3  2
6  5  7  2
2  2  2  0
```

d. *void SubtractMatrix( int matrix1[4][4], int matrix2[4][4], matrix3[4][4] )* 

Your SubtractMatrix function should take in three 4x4 matrices as parameters, and store the difference of the first two matrices into the third. i.e. \( matrix3 = matrix1 - matrix2 \)

Example: For the values of \( matrix1 \) and \( matrix2 \) given in the example under AddMatrix, SubtractMatrix should set
matrix3 =
-3  -3  -3  0
0    1    5   0
2    5    5   0
0    0    0   0

e. void MatrixStats( int matrix1[4][4], float stats[3] )

Your SortMatrix function should take in two 4x4 matrices as parameters, and store the max, min and average of the matrix1 elements in stats[0], stats[1] and stats[2] respectively.

Example: For the value of matrix1 given below

matrix1 =
15  10   8  43
23  17  14   9
39  31  39  15
-8   7   0  -3

the elements in stats[] should be

stat[0] = 43.0
stat[1] = -8.0
stat[2] = 16.0

2. Program Output

In your program, use the GetMatrixElements function to scan in the elements of all the input matrices. Use the PrintMatrix function to print the elements of all the output matrices of AddMatrix and SubtractMatrix. You don’t have to write any special function print values of stats[], you can just use printf.

Your program output for AddMatrix and SubtractMatrix, should look similar to the following.

Enter Matrix1 values
Enter row 0 col 0 value : 1
Enter row 0 col 1 value : 2
...............................
..............................
Enter row 3 col 2 value : 1
Enter row 3 col 3 value : 0

Enter Matrix2 values
Enter row 0 col 0 value : 4
Enter row 0 col 1 value : 5
...............................
..............................
Enter row 3 col 2 value : 1
Enter row 3 col 3 value : 0

The sum of the given matrices is

\[
\begin{pmatrix}
5 & 7 & 9 & 2 \\
4 & 5 & 3 & 2 \\
6 & 5 & 7 & 2 \\
2 & 2 & 2 & 0 \\
\end{pmatrix}
\]

The difference of the given matrices is

\[
\begin{pmatrix}
-3 & -3 & -3 & 0 \\
0 & 1 & 5 & 0 \\
2 & 5 & 5 & 0 \\
0 & 0 & 0 & 0 \\
\end{pmatrix}
\]

Note: Output of GetMatrixElements for the first 2 and last 2 elements, of each input matrix, is only shown here, your output should show this for all values. The output shown above is based on the values of matrix1 and matrix2, shown as an example under AddMatrix.

Enter values of the matrix for which the statistics are wanted

Enter row 0 col 0 value : 15
Enter row 0 col 1 value : 10

..........................

Enter row 3 col 2 value : 0
Enter row 3 col 3 value : -3

The statistics are

Max value : 43.0
Min value : -8.0
Average : -16.0

Note: Output of GetMatrixElements for the first 2 and last 2 elements, of the input matrix, is only shown here, your output should show this for all values. The output shown above is based on the values of matrix1, shown as an example under MatrixStats.

3. What to turn in

Submission for these files will be similar to earlier assignments. You need to create a directory called hw4/. Put all the files listed above in that directory and run the /ecelib/bin/turnin command to submit your homework. You should turn in the following files:

matrix.c
matrix.txt
matrix.script
Use the example matrices given under AddMatrix and MatrixStats sections, and the following matrices, to generate your .script file

Test case 1:
--------------

For AddMatrix and Subtract Matrix

Matrix1 =

\[
\begin{bmatrix}
5 & 6 & 7 & 8 \\
2 & 3 & 4 & 4 \\
-1 & -2 & -1 & 1 \\
1 & 0 & 0 & 0 \\
\end{bmatrix}
\]

Matrix2 =

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

For MatrixStat

Matrix1 =

\[
\begin{bmatrix}
-1 & -3 & -6 & 13 \\
9 & 12 & 11 & 23 \\
15 & 31 & 12 & 39 \\
-8 & -3 & -9 & 100 \\
\end{bmatrix}
\]

Your result should be

For AddMatrix

\[
\begin{bmatrix}
6 & 6 & 7 & 8 \\
2 & 4 & 4 & 4 \\
-1 & -2 & 0 & 1 \\
1 & 0 & 0 & 1 \\
\end{bmatrix}
\]

For SubtractMatrix

\[
\begin{bmatrix}
4 & 6 & 7 & 8 \\
2 & 2 & 4 & 4 \\
-1 & -2 & -2 & 1 \\
1 & 0 & 0 & -1 \\
\end{bmatrix}
\]

For MatrixStat

Max = 100.0
Min = -8.0
Avg = 14.6875

4. Input/Output Redirection

You can use input redirection to avoid typing in 4x4 = 16 values every time you scan in the values of a input matrix
Do the following to reduce your effort during program execution

i. Create a file *input.txt* (can have any name) and enter in all the matrix elements sequentially, each separated by a new line

Example: If you want to scan in two 2x2 arrays say

```
1 1
-2 3
```

and

```
2 3
4 5
```

your input.txt file should look like this

```
1
1
-2
3
2
3
4
5
```

save the *input.txt* file.

ii. Compile your code, assuming your executable is called *matrix_exec*, you can get input values from the file by saying

```
east% ./matrix_exec < input.txt
```

Now whenever *scanf()* is executed in your code, it would get the values from the input.txt file i.e. it would not wait for you to enter a value. This can be done for any program where you have statements waiting for input from the keyboard.

Similarly, output redirection can also be done by using > .

Example:

```
east% ./matrix_exec < input.txt > output.txt
```

Input will be read from *input.txt* file, a file called *output.txt* will be created, and all that is usually printed on the screen without redirection, will be sent to this file.

4. References

Matrix operations: [http://mathworld.wolfram.com/Matrix.html](http://mathworld.wolfram.com/Matrix.html)